3. *Symbolically perturbing the coordinates of each input point (Edelsbrunner, Mücke)*: This is done by adding to each coordinate a suitable power of a small, positive real number represented by a symbolic parameter ϵ. Then, since values are now polynomials in ϵ, the arithmetic operations in the algorithm are replaced by polynomial arithmetic operations. Geometric primitives are implemented symbolically, typically by evaluating a sequence of determinants. Assuming that the dimension of the problem is fixed, this scheme increases the running time of the algorithm by at most a constant factor. However, the overhead incurred can in fact be quite large.

13.8 APPLICATIONS OF GEOMETRY

Geometry overlays the entire computing spectrum, having applications in almost every area of science and engineering, including astrophysics, molecular biology, mechanical design, fluid mechanics, computer graphics, computer vision, geographic information systems, robotics, multimedia, and mechanical engineering.

The growing availability of large geometric databases is a major driver of the increase in applications. GIS mapping databases containing most of the world's roads enable route planning applications. Airborne LIDAR creates terrain elevation databases. That enables observer visibility computation, with applications ranging from radio tower siting to visual nuisance mitigation. Laser scanners produce 3D point clouds recording the surfaces of objects from buildings down to sculptures and even people. The ensuing applications include deducing the structure of those objects' surfaces.

13.8.1 MATHEMATICAL PROGRAMMING

Definition:

Mathematical programming is the large-scale optimization of an objective function (such as cost) of many variables subject to constraints, such as supplies and capacities.

Facts:

1. Mathematical programming includes both *linear programming* (continuous, integer, and network) and *nonlinear programming* (quadratic, convex, general continuous, and general integer).

2. Applications include transportation planning and transshipment, factory production scheduling, and even determining a least-cost but adequate diet.

3. A *modeling language*, such as AMPL, is often used in applications [FoGaKe93].

Example:

1. *Linear programming in low dimensions*: This is a special case of linear programming since there are algorithms whose time is linear in the number of constraints but exponential in the number of dimensions.

Application:

1. Find the *smallest enclosing ball* of a set of points or a set of balls in arbitrary dimension [We91]. This uses a randomized incremental algorithm employing the move-to-front heuristic.

13.8.2 POLYHEDRAL COMBINATORICS

Polyhedra have been important to geometry since the classification of regular polyhedra in classical times.

Fact:

1. The following are some of the many possible operations that can be performed on polygons and polyhedra:

- *Boolean operations*, such as intersection, union, and difference;
- *point location* of new query points in a preprocessed set of polygons, which may partition a larger region;
- *range search* of a preprocessed set of points to find those inside a new query polygon;
- *decomposition* (or *triangulation*) of a polygon into triangles or a polyhedron into tetrahedra (§13.4.2). A simple n-gon is always decomposable into $n-2$ triangles, in linear time, and all triangulations have exactly $n-2$ triangles. However, some polyhedra in $\mathcal{R}^3$ cannot be partitioned intro tetrahedra without additional Steiner points. Also, different triangulations of the same polyhedron may have different numbers of tetrahedra.

Applications:

1. *Aperiodic tilings and quasicrystals*: Tilings (in the plane) and crystallography (in 3-dimensional space) are classic applications of polygons and polyhedra. A recent development is the study of aperiodic (Penrose) tilings [Ga77] and quasicrystals [Ap94]. These are locally but not globally symmetric under 5-fold rotations, quasi-periodic with respect to translations, and self-similar. The 1991 Nobel Prize in Chemistry was awarded for the discovery of actual quasicrystals:

- `http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2011/`

Large-scale symmetries, such as 5-fold, that are impossible in traditional crystallography, can be visible with X-ray diffraction. Tilings can be constructed by projecting simple objects from, say, $\mathcal{R}^5$. One application is the surface reinforcement of soft metals.

2. *Error-correcting codes*: Some error-correcting codes can be visualized with polytopes as follows. Assume that the goal is k-bit symbols, where any error of up to b bits can be detected. The possible symbols are some of the 2^k vertices of the hypercube in k-dimensional space. The set of symbols must contain no two symbols less than $b+1$ distance apart, where the metric is the number of different bits. If errors of up to c bits are to be correctable, then no two symbols can be closer than $2c+1$.

Similarly, in quantum computing, a *quantum error-correcting code* can be designed using Clifford groups and binary orthogonal geometry [CaEtal97].

13.8.3 COMPUTATIONAL CONVEXITY

Definitions:

An ***$\mathcal{H}$-polytope*** is a polytope defined as the intersection of m half-spaces in $\mathcal{R}^n$.

A ***$\mathcal{V}$-polytope*** is a polytope defined as the convex hull of m points in $\mathcal{R}^n$.

A ***zonotope*** is the vector (Minkowski) sum of a finite number of line segments.

Computational convexity is the study of high-dimensional convex bodies.

An ***oracle*** is an algorithm that gives information about a convex body.

Facts:

1. Computational complexity is related to linear programming, polyhedral combinatorics, and the algorithmetic theory of polytopes and convex bodies.

2. In contrast to computational geometry, computational complexity considers convex structures in normed vector spaces of finite but not restricted dimension. If the body under consideration is more complex than a polytope or zonotope, it may be represented as an oracle. Here the body is a black-box, and all information about it, such as membership, is supplied by calls to the oracle function. Typical algorithms involve *volume computation*, either deterministically, or by *Monte Carlo* methods, perhaps after decomposition into simpler bodies, such as simplices.

3. When the dimension n is fixed, the volume of $\mathcal{V}$-polytopes and $\mathcal{H}$-polytopes can be computed in polynomial time.

4. There does not exist a polynomial-space algorithm for the exact computation of the volume of $\mathcal{H}$-polytopes (where n is part of the input).

5. Additional information on computational convexity can be found at

- `http://dimacs.rutgers.edu/TechnicalReports/TechReports/1994/94-31.ps`

13.8.4 MOTION PLANNING IN ROBOTICS

In Computer Assisted Manufacturing, both the tools and the parts being assembled must often be moved around each other in a cluttered environment. Their motion should be planned to avoid collisions, and then to minimize cost.

Definition:

A ***Davenport-Schinzel sequence*** of order s over an alphabet of size n, or $DS(n,s)$, is a sequence of characters such that:

- no two consecutive characters are the same;
- for any pair of characters, a and b, there is no alternating subsequence of length $s+2$ of the form $\ldots a \ldots b \ldots a \ldots b \ldots$.

Facts:

1. Practical general motion, *path planning*, is solvable with Davenport-Schinzel sequences [ShAg95]. Upper bounds on $\lambda_s(n)$, the length of the longest $DS(n,s)$, determine upper bounds on the complexity of the lower envelopes of certain functions.

For example, given n points in the plane that are moving with positions that are polynomials of degree s in time, the number of times that the closest pair of points can change is $\lambda_{2s}(\binom{n}{2})$.

2. Visibility graphs (§13.5.5) are useful in finding a shortest path between two points in the plane, in the presence of obstacles.

3. The problem of moving a finite object in the presence of obstacles may also be mapped into a *configuration space* (or *C-space*) problem of moving a corresponding point in a higher dimension. If translational and rotational motion in 2-dimensional (respectively 3-dimensional) is allowed, then the C-space is 3-dimensional (respectively 6-dimensional).

4. Articulated objects, multiple simultaneous motion, and robot hands also increase the number of degrees of freedom.

5. Current problems:

- *representation* of objects, since although planar, faceted models are simpler, the objects should be algebraic surfaces, and even if they are planar, in C-space their corresponding versions will be curved;
- *grasping*, or placing a minimal number of fingers to constrain the object's motion;
- *sequence planning* of the assembly of a collection of parts;
- *autonomous navigation* of robots in unstructured environments.

13.8.5 CONVEX HULL APPLICATIONS

Facts:

1. The convex hull (§13.5.1) is related to the Voronoi diagram (§13.5.3) since a convex hull problem in $\mathcal{R}^k$ is trivially reducible to a Voronoi diagram problem in $\mathcal{R}^k$, and a Voronoi diagram problem in $\mathcal{R}^k$ is reducible to a convex hull problem in $\mathcal{R}^{k+1}$.

2. The definition of convex hull is not constructive, in that it does not lead to a method for finding the convex hull. Nevertheless, there are many constructive algorithms and implementations. One common implementation is *QuickHull* (§13.5.1), a general dimension code for computing convex hulls, Delaunay triangulations, Voronoi vertices, furthest-site Voronoi vertices, and half-space intersections [BaDoHu96].

3. *Alpha-shapes* are a useful generalization of convex hulls [Ed05]. An intuitive definition is as follows. For a given α, form a surface by rolling around the given point set with a sphere of radius α, then smooth out the curved regions. The specification $\alpha = \infty$ gives the convex hull of the point set. The smaller the value of α, the more closely the original points are followed. One application is inferring a surface from a 3D point set. A good introduction to the topic is found at

- `http://graphics.stanford.edu/courses/cs268-16-fall/Handouts/AlphaShapes/as_fisher.pdf`

Examples:

1. *Mathematics*:

- determining the principal components of spectral data;
- studying circuits of matroids that form a Hilbert base;
- studying the neighbors of the origin in the $\mathcal{R}^8$ lattice.

2. *Biology and medicine*:

- classifying molecules by their biological activity;
- determining the shapes of left ventricles for electrical analysis of the heart.

3. *Engineering*:

- computing support structures for objects in layered manufacturing in rapid prototyping [StBrEa95]. By supporting overhanging material, these structures prevent the object from toppling while partially built.
- designing nonlinear controllers for controlling vibration;
- finding invariant sets for delta-sigma modulators;
- classifying handwritten digits;
- analyzing the training sets for a multilayer perceptron model;
- determining the operating characteristics of process equipment;
- navigating robots;
- creating 6-dimensional wrench spaces to measure the stability of robot grasps;
- building micromagnetic models with irregular grain structures;
- building geographical information systems;
- simulating a spatial database system to evaluate spatial tesselations for indexing;
- producing virtual reality systems;
- performing discrete simulations of incompressible viscous fluids using vortex methods;
- modeling subduction zones of tectonic plates and studying fluid flow and crystal deformation;
- computing 3-dimensional unstructured meshes for computational fluid dynamics.

13.8.6 NEAREST NEIGHBOR

Variants of the problem of finding the nearest pair of a set of points have applications in fields from handwriting recognition to astrophysics.

Facts:

1. *Fixed search set, varying query point*: A fixed set $\mathcal{P}$ of n points in $\mathcal{R}^d$ is preprocessed so that the closest point $p \in \mathcal{P}$ can be found for each query point q. The search time per query can range from $\log n$ (if $d = 2$) to $n^{\frac{d}{2}}$ (for large d). The Voronoi diagram is commonly used in low dimensions. However, because of the Voronoi diagram's complexity in higher dimensions, hierarchical search structures, bucketing, and probabilistic methods perhaps returning approximate answers are common.

2. *Moving points*: The points in $\mathcal{P}$ may be moving and the close pairs of points over time is of interest.

Examples:

1. Fixed search sets, varying query point:

- *Character recognition in document processing*: Each representative character is defined by a vector of features. Each new, unknown character must be mapped to the closest representative character in feature space.
- *Color map optimization in computer graphics*: Many frame buffers allow only the 256 colors in the current color map to be displayed simultaneously, from a palette of 2^{24} possible colors. Thus, each color in a new image must be mapped to the closest color in the color map. A related problem is the problem of determining what colors to use in the color map.

- *Clustering algorithms for speech and image compression in multimedia systems*: As in the color map problem, a large number of points must be quantized down to a smaller set.

2. Moving points:

- *Simulation of star motion in astrophysics*: Calculating the gravitational attraction between every pair of stars is too costly, so only close pairs are individually calculated. Otherwise the stars are grouped, and the attraction between close groups is calculated. The groups may themselves be grouped hierarchically.
- *Molecular modeling*: In molecular modeling, close pairs of atoms will be subject to van der Waals forces.
- *Air traffic control*: Air traffic controllers wish to know about pairs of aircraft closer than a minimum safe distance. Here the metric is nonuniform; small vertical separations are more tolerable than small horizontal separations.
- During path planning in robotics and numerically controlled machining, unintended close pairs of objects must also be avoided.

13.8.7 COMPUTER GRAPHICS

Computer graphics may be divided into *modeling* of surfaces, and *simulation* of the models. The latter includes *rendering* a scene and its light sources to generate synthetic imagery with respect to some viewpoint. Rendering involves *visibility*, or determining which parts of the surfaces are visible, and *shading* them according to some lighting model.

Definitions:

Anti-aliasing refers to filtering out high-frequency spatial components of a signal, to prevent artifacts, or aliases, from appearing in the output image. In graphics, a high frequency may be an object whose image is smaller than one pixel or a sharp edge of an object.

A ***GUI*** (***graphical user interface***) is a mechanism that allows a user to interactively control a computer program with a bitmapped display by using a mouse or pointer to select menu items, move sliders or valuators, and so on. The keyboard is only occasionally used. A GUI contrasts with typing the program name followed by options on a command line, or by preparing a text file of commands for the program. A GUI is easier and more intuitive to use, but can slow down an expert user.

Examples:

1. *Visibility*: Visibility algorithms may be *object-space*, where the visible parts of each object are determined, or *image-space*, where the color of each pixel in the frame buffer is determined. The latter is often simpler, but the output has less meaning, since it is not referred back to the original objects. Techniques include *ray tracing* and *radiosity*.

- *Ray tracing*: Ray tracing extends a line from viewpoint through each pixel of the frame buffer until the first intersecting object. If that surface is a mirror, then the line is reflected from the surface and continues in a different direction until it hits another object (or leaves the scene). If the object is glass, then both a reflecting and a refracting line are continued, with their colors to be combined according to Fresnel's law.

One geometry problem here is that of *sampling for subpixel averaging*. The goal is to color a square pixel of a frame buffer according to the fraction of its area occupied by each visible object. Given a line diagonally crossing a pixel, the fraction of the pixel covered by that face must be obtained for anti-aliasing. If the edges of two faces intersect in this pixel, each face cannot be handled independently, for example with an anti-aliased Bresehnam algorithm. If this is done badly, then it is very obvious in the final image as a possible fringe of a different color around the border of the object [Mi96].

The solution is to pick a small set of points in the pixel (typically 9, 16, or 64 points), determine which visible object projects to each point, and combine those colors. The problem is then to select a set of sampling points in the pixel, such that given a subset region, the number of points in it approximates its area. Four possible methods, from worst to best, are to

- ◇ pick the points independently and uniform randomly;
- ◇ use a nonrandom uniform distribution;
- ◇ start with the above distribution, then jitter the points, or perturb each one slightly;
- ◇ use simulated annealing to improve the point distribution.

- *Radiosity*: Radiosity partitions the scene into facets, computes a *form factor* of how much light from each facet will impinge on each other, and solves a system of linear equations to determine each facet's brightness. This models diffuse lighting particularly well.
- *Windowing systems*: Another visibility problem is designing the appropriate data structure for representing the windows in a GUI, so that the window that is in front at any particular pixel location can be determined, in order to receive the input focus.
- *Radio wave propagation*: The transmission of radio waves, as from cellular telephones, which are reflected and absorbed by building contents, is another application of visibility [Fo96].

2. *Computer vision*: Applications of geometry to vision include *model-based recognition* (or *pattern matching*), and *reconstruction or recovery of 3-D structure* from 2-D images, such as stereopsis, and structure from motion. In recognition, a model of an object is transformed into a sensor-based coordinate system and the transformation must be recovered. In reconstruction, the object must be determined from multiple projections.

3. *Medical image shape reconstruction*: Various medical imaging methods, such as computer tomography, produce data in the form of successive parallel slices through the body. The basic step in reconstructing the 3-dimensional object from these slices in order to view it involves joining the corresponding vertices and edges of two polygons in parallel planes by triangles to form a simple polyhedron. However, there exists a pair of polygons that cannot be so joined [GiORSu96].

13.8.8 MECHANICAL ENGINEERING DESIGN AND MANUFACTURING

Geometry is very applicable in CAD/CAM, such as in the design and manufacture of automobile bodies and parts, aircraft fuselages and parts such as turbine blades, and ship hulls and propellers. The current leading edge of CAD/CAM is ***additive manufacturing***, also known as ***3D printing***.

Examples:

1. *Representations*: How should mechanical parts be represented? One problem is that geometric descriptions are verbose compared to 2-dimensional descriptions, such as draftings, since those assume certain things that the users will fill in as needed, but which must be explicit in the geometric description. Currently, it is possible to additively manufacture objects that cannot be represented so that their properties can be analyzed. The problem is objects containing lattices with billions of cells and nonhomogeneous (graded) materials. See George Allen's comments in

- `http://www.3dcadworld.com/why-cad-is-hard-geometric-problems/`
- `http://www.cs.technion.ac.il/gdm2014/Presentations/GDM2014_allen.pdf`

Possible representation methods include the following:

- *constructive solid geometry*: Primitive objects, such as cylinders and blocks, are combined with the regularized Boolean operators union, intersection, and difference.
- *faceted boundary representation*: The object is a polyhedron with a boundary of planar faces.
- *exact boundary representation*: The object is defined by boundary "faces", but now each face can be curved, such as a NURBS (Non-Uniform Rational B-Spline), or an implicit piecewise quadric, Dupin cyclide (a quartic surface that is good for blending two quadric surfaces), or supercyclide.

The possible methods can be evaluated with the following criteria:

- *robustness* against numerical errors;
- *elegance*;
- *accuracy* in representing complex, curved, shapes, especially blends between the two surfaces at the intersection of two components;
- *ease* of explicitly obtaining geometry such as the boundary;
- *efficiency* when implemented on GPUs, which prefer simple, regular, representations.

2. *Mesh generation*: A *mesh* is the partition of a polyhedron into, typically, tetrahedra or hexahedra to facilitate finite element modeling. A good mesher conforms to constraints, can change scale over a short distance, has no unnecessary long thin elements, and has fewer elements when possible. In some applications, periodic *remeshing* is required.

If the elements are tetrahedra, then a Delaunay criterion that the circumsphere of each tetrahedron contains no other vertices may be used. However, this is inappropriate in certain cases, such as just exterior to an airfoil, where a (hexahedral) element may have an aspect ratio of 100,000:1. This raises numerical computation issues.

Applications of meshing outside mechanical design include *computational fluid dynamics*, *contouring* in GIS, *terrain databases* for real-time simulations, and Delaunay applications in general. Some mesh generation and additive manufacturing programs include the following.

- *Triangle*, a 2-dimensional quality mesh generator and Delaunay triangulator. Triangle generates exact Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams, and triangular meshes. The latter can be generated with no small or large angles, and are thus suitable for finite element analysis.

`https://www.cs.cmu.edu/~quake/triangle.html`

- *TetGen*, a quality tetrahedral mesh generator and a 3D Delaunay triangulator.

  `http://wias-berlin.de/software/tetgen/`

- *MeshLab* is an open-source system for processing and editing 3D triangular meshes. It provides a set of tools for editing, cleaning, healing, inspecting, rendering, texturing, and converting meshes. It offers features for processing raw data produced by 3D digitization tools/devices and for preparing models for 3D printing.

  `http://www.meshlab.net/`

- *ImplicitCAD* is an open-source, programmatic CAD environment.

  `http://www.implicitcad.org/`

- *Autodesk Netfabb* is an excellent commercial product with a comprehensive range of connected tools, including design optimization and printing.

  `https://www.netfabb.com/`

- *Slic3r* is a widely used free open and flexible toolchain for 3D printers. It introduced many features such as multiple extruders, brim, microlayering, bridge detection, command line slicing, variable layer heights, sequential printing (one object at time), honeycomb infill, mesh cutting, object splitting into parts, AMF support, avoiding crossing perimeters, and distinct extrusion widths.

  `http://slic3r.org/`

3. *Minimizing workpiece setup in numerically controlled (NC) machining*: In 4- and 5-axis numerically controlled machining, in order to machine all the faces, the workpiece must be repeatedly dismounted, recalibrated, and remounted. This setup can take much more time than the actual machining. Minimizing the number of setups by maximizing the number of faces that can be machined in one setup is a visibility problem harder than finding an optimal set of observers to cover some geographic terrain. Exact solutions are NP-hard; approximate solutions use geometric duality, topological sweeping, and efficient construction and searching of polygon arrangements on a sphere.

4. *Dimensional tolerancing*: *Tolerancing* refers to formally modeling the relationships between mechanical function and geometric form while assigning and analyzing dimensional tolerances to ensure that parts assemble interchangeably [SrVo93]. A tolerance may be specified *parametrically*, as a variation in a parameter, such as the width of a rectangle, or as a *zone* that the object's boundary must remain in. The latter is more general but must be restricted to prohibit pathologies, such as the object's boundary being not connected.

Tolerance synthesis attempts to optimize the tolerances to minimize the manufacturing cost of an object, considering that, while large tolerances are cheaper to manufacture, the resulting product may function poorly [Sk96].

Unsolved Problems:

The following problems are perpetually on the list of those needing more research.

1. *Blending* between two surfaces in mechanical design, especially at the ends of the blend, where these surfaces meet others. (A *blending surface* smooths the intersection of two surfaces by being tangent to them, each along a curve.)

2. *Variational design* of a class of objects subject to constraints. Well-designed constraint systems may have multiple solutions; the space must be searched for the correct one. Labeling derivative entities, such as the edge resulting from the intersection of two inputs, is an issue partly because this edge may not exist for some parameter values.

3. Generally *formalizing the semantics* of solid modeling [Ho96].

4. Updating *simplifying assumptions*, such as the linearity of random access memory, and points being in general position, which were useful in the past, but which cause problems now.

5. *Accounting for dependencies* between geometric primitives, and *maintaining topological consistency*.

6. Designing *robust algorithms* when not only is there numerical roundoff during the computation, but also the input data are imprecise, for example, with faces not meeting properly.

7. Better *3-dimensional anti-aliasing* to remove crevices and similar database errors before *rapid prototyping*.

8. There still remains a need for many features in geometry implementations, such as more *geometric primitives* at all levels, *default visualization or animation* easily callable for each data structure, more *rapid prototyping* with visualization, a *visual debugger* for geometric software, including changing objects online, and generally *more interactivity*, not just data-driven programs.

13.8.9 LAYOUT PROBLEMS

The efficient layout of objects has wide-ranging applications in geometry.

Examples:

1. *Textile part layout*: The clothing industry cuts parts from stock material after performing a tight, nonoverlapping, layout of the parts, in order to minimize the costs of expensive material. Often, because the cloth is not rotationally symmetric, the parts may be translated, but not rotated. Therefore, geometric algorithms for minimizing the overlap of translating polygons are necessary. Since this problem is PSPACE-hard, heuristics must be used [Da95], [LiMi95].

2. *VSLI layout*: Both laying out circuits and analyzing the layouts represent important problems. The masks and materials of a VLSI integrated circuit design are typically represented as rectangles, mostly isothetic, although 45 degrees or more general angles of inclination for the edges are becoming common. The rectangles of different layers may overlap. One integrated circuit may be 50MB of data before its hierarchical data structure is flattened, or 2GB after. For further details, see [WeHa11].

Geometry problems include the following.

- *design rule verification*: It is necessary to check that objects are separated by the proper distances and that average metal densities are appropriate for the fabrication process.
- *polygon simplification*: A design described by a complex set of polygons may perhaps be optimized into a smaller set of isothetic polygons (with only horizontal and vertical sides), such that the symmetric difference from the original design is as small as possible.

- *logic verification*: The electrical *circuit* is determined by the graph extracted from the adjacency information of the rectangles, and whether it matches the original logic design is determined. A subproblem is determining *devices* (*transistors*), which occur when rectangles of two particular different layers overlap.
- *capacitance*: This depends on the closeness of the component rectangles, which might be overlapping or separated, representing two conductors.
- *PPC* (Process Proximity Correction): This means to correct the effect that, when *etching* a circuit, a rectangle's edges are displaced outward, possibly causing it to come too close to another rectangle, and change the circuit.

13.8.10 GRAPH DRAWING AND VISUALIZATION

The classic field of *graph drawing and visualization* (see [KaWa01], [Ta14], and [TaTo95]) aims automatically to display a graph, emphasizing fundamental properties such as symmetry while minimizing the ratio between longest and shortest edges, number of edge crossings, etc. Applications include advanced GUIs, visualization systems, databases, showing the interactions of individuals and groups in sociology, illustrating connections between components in software engineering, and visualization of biological networks, including phylogenetic trees, metabolic networks, protein-protein interaction networks, and gene regulatory networks.

Facts:

1. Graph G can be drawn as the 1-skeleton of a convex polytope in $\mathcal{R}^3$ if and only if G is planar and 3-connected (Steinitz). See [Gr67].

2. Given a 3-connected planar graph, the graph can be drawn as a convex polyhedron in $\mathcal{R}^3$ using $O(n)$ volume while requiring the vertices to be at least unit distance apart, which allows them to be visually distinguished. This can be done in $O(n^{1.5})$ time [ChGoTa96].

3. A special language GraphML is available for representing graphs and drawings of these graphs.

4. Three of the most widely used software systems for constructing graph drawings are the Open Graph Drawing Framework (OGDF), the GDTookit, and the Public Implementation of a Graph Algorithm Library and Editor PIGALE, available at

- `http://www.ogdf.net`
- `http://www.dia.uniroma3.it/~gdt`
- `http://pigale.sourceforge.net`

5. There are a variety of hardware and software systems available for visualization of 3-dimensional graph drawings. See [La01] and the conference sites

- `http://algo.math.ntua.gr/~gd2016/`
- `http://www.diagrams-conference.org/2016/`

13.8.11 GEOGRAPHIC INFORMATION SYSTEMS

A *map* (§8.6.4) is a planar graph. Minimally, it contains *vertices*, *edges*, and *polygons*. However, a sequence of consecutive edges and 2-vertices is often called a *chain* (or *polyline*), and its interior vertices *points*. For example, if each polygon is one nation, then the southern border of Canada with the USA is one chain.

Definition:

A ***geographic information system*** (***GIS***) is an information system designed to capture, store, manipulate, analyze, and display spatial or geographically-referenced data.

Facts:

Typical simple geometric operations are given in Facts 1–6. More complex ones are given in Facts 7–9.

1. *Projecting data from one map projection to another, and determining the appropriate projection*: Since the earth is not a developable surface, no projection meets all the following criteria simultaneously: *equal-area*, *equidistant* (preserving distances from one central point to every other point), *conformal* (preserving all angles), and *azimuthal* (correctly showing the compass angle from one central point to every other point). Since a projection that meets any one criterion exactly is quite bad in the others, the most useful projections tend to be compromises, such as the recent *Robinson projection* [Da95].

2. *Rubber-sheeting*, or nonlinear stretching, to align a map with calibration points, and for *edge joining* of adjacent map sheets or databases, which may have slightly different coordinate systems.

3. *Generalizing* or reducing the number of points in a chain while preserving certain error properties.

4. *Topological cleanup* so that edges that are supposed to meet at one vertex do so, the boundary of each polygon is a closed sequence of vertices and polylines, adjacency information is correct, and so on.

5. Choice of the correct *data structure*. Should elevation data be represented in a *gridded* form (as an array of elevations) or should a *triangulated irregular network* (TIN) be used (the surface is partitioned into triangles)?

6. *Zone of influence calculation*: For example, find all the national monuments within ten miles of the highway.

7. *Overlaying*: Overlaying two maps to produce a third, where one polygon of the overlay map will be those points that are all from the same two polygons of the two input maps is one of the most complex operations in a GIS. If only the *area* or other mass property of the overlay polygons is desired, then it is not necessary completely to find the overlay polygons first; it is sufficient to find the set of vertices and their neighborhoods of each overlay polygon [FrEtal94].

8. *Name placement*: Consider a cartographic map containing point features such as cities, line features such as rivers, and area features such as states. The *name placement* problem involves locating the features' names so as to maximize readability and aesthetics [FrAh84]. Efficient solutions become more important as various mapping packages now produce maps on demand. The techniques also extend to *labelling CAD drawings*, such as piping layouts and wiring diagrams.

9. *Viewsheds and visibility indices*: Consider a terrain database, and an observer and target, both of which may be some distance above the terrain. The observer can see the target if and only if a line between them does not intersect the terrain. Note that if they are at different heights above the terrain, then this relation is not necessarily commutative.

The (not necessarily connected) polygon of possible targets visible by a particular observer is his *viewshed*. The viewshed's area is the observer's *visibility index*. In order to site observers optimally, the visibility index for each possible observer in the database may be required. Calculating this exactly for an $n \times n$ gridded database takes time $O(n^5)$ so sampling techniques are used [FrRa94].

Implementations:

1. *GRASS GIS*, also known as GRASS (Geographic Resources Analysis Support System), is a free and open-source Geographic Information System (GIS) software suite widely used used for geospatial data management and analysis, image processing, graphics and maps production, spatial modeling, and visualization.

- `https://grass.osgeo.org/`

2. *The Open Source Geospatial Foundation* (OSGeo) was created to support the collaborative development of open-source geospatial software and to promote its widespread use.

- `http://www.osgeo.org/`

13.8.12 GEOMETRIC CONSTRAINT SOLVING

Applications of geometric constraint solving include mechanical engineering, molecular modeling, geometric theorem proving, and surveying.

Definition:

Geometric constraint solving is the problem of locating a set of geometric elements given a set of constraints among them.

Fact:

1. The problem may be under-constrained, with an infinite number of solutions, or over-constrained, with no solutions without some relaxation.

Examples:

1. A *receptor* is a rigid cavity in a protein, which is the center of activity for some reaction. A *ligand* is a small molecule that may bind at a receptor. The activity level of a drug may depend on how the ligand fits the receptor, which is made more complicated by the protein molecule's bending.

2. In CAD/CAM, there may be constraints such as the specification that opposite sides of a feature must be parallel. For example, commercial systems like Pro/Engineer allow the user to freehand-sketch a part, and then apply constraints such as right angles, to snap the drawing to fit. Then the user is required to add more constraints until the part is well constrained.

3. *Molecular modeling*: There is often a lock-and-key relationship between a flexible protein molecule's receptor and the ligand that it binds. In addition to geometrically matching the fitted shapes, the surface potentials of the molecules is also important. This fitting problem, called *molecular docking*, is important in *computer-aided drug design*. Generally, a heuristic strategy is used to move the molecules to achieve no overlap between the two molecules while maximizing their contact area [IePa95].

13.8.13 IMPLEMENTATIONS

One major application of geometry is in implementations of geometric software packages, either as *class libraries* and *subroutine packages* callable from user programs, or as *stand-alone systems*, which the user prepares input data files for and directs with either input command files or a GUI. These implementations are useful both for creating or modeling geometric objects and for visualizing them. This section describes some implementations of general usefulness across several applications.

Definitions:

In an object-oriented computer language, a ***class library*** is a set of new data types (classes) and operations on them, activated by sending a message to an ***object***, or data item. (For example, a plane object may respond to a message to rotate itself. The internal representation of an object is private, and it may be accessed only by sending it a message.)

In C++, class libraries often use ***template metaprogramming***. Here, class templates are expanded at compile time into code that often runs much more efficiently than traditional subroutine libraries. This trades off compilation time for execution time, especially for complicated classes with efficient specialized cases. For example, consider a matrix that is known at compile time to be of size 3×3, which is common in geometric applications. Here, specialized code is generated for any loop iterating over the rows of the matrix, probably using loop unrolling. Even better, expressions with several matrix operations chained together become code with no temporary matrix storage allocations, and with all the code located together in one routine and available for global optimization.

Example libraries:

1. *Boost.Geometry* (also known as the Generic Geometry Library, GGL) is a part of the collection of Boost C++ Libraries. It defines concepts, primitives, and algorithms for solving geometry problems.

- `http://www.boost.org/doc/libs/1_63_0/libs/geometry/doc/html/geometry/introduction.html`

Boost.Geometry contains a dimension-agnostic, coordinate-system-agnostic and scalable kernel, based on concepts, meta-functions, and tag dispatching. Supported algorithms include area, length, perimeter, centroid, convex hull, intersection (clipping), within (point in polygon), distance, envelope (bounding box), simplify, and transform. Boost.Geometry can be used wherever geometry plays a role, such as in mapping and GIS, game development, computer graphics and widgets, robotics, and astronomy. However, the development has been mostly GIS-oriented.

2. *Computational Geometry Algorithms Library* (CGAL) is a software project that provides access to efficient and reliable geometric algorithms in the form of a C++ library. CGAL is used in various areas needing geometric computation, such as geographic information systems, computer aided design, molecular biology, medical imaging, computer graphics, and robotics.

The library offers data structures and algorithms like triangulations, Voronoi diagrams, Boolean operations on polygons and polyhedra, point set processing, arrangements of curves, surface and volume mesh generation, geometry processing, alpha shapes, convex hull algorithms, shape analysis, AABB and KD trees, among others.

CGAL is a massive project that started in 1996 as a consortium of several existing projects. It was originally funded by the European Union's information technologies program ESPRIT.

- `http://www.cgal.org/`

3. *Geogram* is a C++ programming library of geometric algorithms, with efficient graphics for surfacic and volumetric meshes.

- `http://homepages.loria.fr/BLevy/GEOGRAM/`

Example stand-alone systems:

1. *Wolfram|Alpha* is an engine for carrying out dynamic computations, searching for answers, and providing knowledge, based on the *Mathematica* system. Alpha can answer questions such as "How many baseballs fit in a Boeing 747?", where the answer makes a reasonable assumption about the packing density, and states it. More technical questions, such as intersecting lines, computing properties of polytopes, and drawing fractals, are also possible.

- `https://www.wolframalpha.com/about.html`
- `https://www.wolfram.com/mathematica/`

2. *Blender* is a free and open-source 3D creation suite. It supports modeling, rigging, animation, simulation, rendering, compositing and motion tracking, video editing, and game creation.

- `https://www.blender.org/`

3. Several very efficient geometry and GIS programs for processing large datasets are available at

- `www.ecse.rpi.edu/Homepages/wrf/Software`

Here is a sampling of the supported software:

- *PinMesh* is a very fast algorithm with the facility to preprocess a polyhedral mesh in order to perform 3D point location queries [MaEtal16].
- *TiledVS* allows one to efficiently compute terrain viewsheds on large raster terrains using small amounts of real memory with a custom virtual memory manager [FeEtal16].
- *UPLAN* is an efficient algorithm for path planning on road networks having polygonal constraints [MaEtal15b].
- *EMFlow* and *RWFlood* efficiently compute hydrography (water flow) on very large (50000×50000) terrains containing basins [GoEtal15]; the novel component was to raise the ocean and compute how it flowed over sills to flood interior basins.
- *EPUG-Overlay* allows one to compute the overlay of two GIS maps (plane graphs) in parallel using rational numbers, with the ability to process maps with tens of millions of vertices and hundreds of thousands of polygons in a few minutes [MaEtal15a].
- *NearptD* is a very fast parallel nearest neighbor algorithm and implementation, which has processed 10^7 points in E^6 and $184 \cdot 10^6$ points in E^3.

REFERENCES

Printed Resources:

[AgEtal96] P. K. Agarwal, B. Aronov, J. Pach, R. Pollack, and M. Sharir, "Quasi-planar graphs have a linear number of edges", in *Graph Drawing '95*, Lecture Notes in Computer Science 1027, Springer-Verlag, 1996, 1–7.

[AjEtal82] M. Ajtai, V. Chvátal, M. M. Newborn, and E. Szemerédi, "Crossing-free subgraphs", *Annals of Discrete Mathematics* 12 (1982), 9–12.

[AkAl89] J. Akiyama and N. Alon, "Disjoint simplices and geometric hypergraphs", *Combinatorial Mathematics*, Annals of the New York Academy of Sciences 555 (1989), 1–3.

[AlKl92] N. Alon and D. Kleitman, "Piercing convex sets and the Hadwiger-Debrunner (p, q)-problem", *Advances in Mathematics* 96 (1992), 103–112.

[Al63] E. Altman, "On a problem of Erdős", *American Mathematical Monthly* 70 (1963), 148–157.

[Ap94] *Aperiodic 1994, International Conference on Aperiodic Crystals*, Les Diablerets, Switzerland, 1994.

[ArEtal91] B. Aronov, B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and R. Wenger, "Points and triangles in the plane and halving planes in space", *Discrete & Computational Geometry* 6 (1991), 435–442.

[AtGo93] M. J. Atallah and M. T. Goodrich, "Deterministic parallel computational geometry", in *Synthesis of Parallel Algorithms*, J. H. Reif (ed.), Morgan Kaufmann, 1993, 497–536.

[BaKe92] A. Bachem and W. Kerns, *Linear Programming Duality, An Introduction to Oriented Matroids*, Springer-Verlag, 1992.

[BaDe92] C. Bajaj and T. K. Dey, "Convex decomposition of polyhedra and robustness", *SIAM Journal on Computing* 21 (1992), 339-364.

[Ba91] K. Ball, "The plank problem for symmetric bodies", *Inventiones Mathematicae* 104 (1991), 535–543.

[Bá82] I. Bárány, "A generalization of Carathéodory's theorem", *Discrete Mathematics* 40 (1982), 141–152.

[BáFüLo90] I. Bárány, Z. Füredi, and L. Lovász, "On the number of halving planes", *Combinatorica* 10 (1990), 175–183.

[BáKaPa82] I. Bárány, M. Katchalski, and J. Pach, "Quantitative Helly-type theorems", *Proceedings of the American Mathematical Society* 86 (1982), 109–114.

[BaDoHu96] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, "The Quickhull algorithm for convex hulls", *ACM Transactions on Mathematical Software* 22 (1996), 469–483.

[Be83] J. Beck, "On the lattice property of the plane and some problems of Dirac, Motzkin and Erdős in combinatorial geometry", *Combinatorica* 3 (1983), 281–297.

[Be10] K. Bezdek, *Classical Topics in Discrete Geometry*, CMS Books in Mathematics, Springer, 2010.

[Be13] K. Bezdek, *Lectures on Sphere Arrangements – the Discrete Geometric Side*, Fields Institute Monographs, Volume 32, Springer, 2013.

[BeKu90] A. Bezdek and W. Kuperberg, "Examples of space-tiling polyhedra related to Hilbert's Problem 18, Question 2", in *Topics in Combinatorics and Graph Theory*, R. Bodendiek and R. Henn (eds.), Physica-Verlag, 1990, 87–92.

[BeLá15] K. Bezdek and Zs. Lángi, "Density bounds for outer parallel domains of unit ball packings", *Proceedings of the Steklov Institute of Mathematics* 288 (2015), 209–225.

[BjEtal93] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler, *Oriented Matroids*, Cambridge University Press, 1993. (A comprehensive monograph on the theory of oriented matroids.)

[Bo93] J. Bokowski, "Oriented matroids", in *Handbook of Convex Geometry, Vol. A*, P. M. Gruber and J. M. Wills (eds.), North Holland, 1993, 555–602.

[BoSt89] J. Bokowski and B. Sturmfels, *Computational Synthetic Geometry*, Lecture Notes in Mathematics 1355, Springer-Verlag, 1989.

[BoFü84] E. Boros and Z. Füredi, "The number of triangles covering the center of an n-set", *Geometria Dedicata* 17 (1984), 69–77.

[Br96] P. Brass, "Erdős distance problems in normed spaces", *Computational Geometry* 6 (1996), 195–214.

[BuGrSl79] S. Burr, B. Grünbaum, and N. Sloane, "The orchard problem", *Geometria Dedicata* 2 (1979), 397–424.

[CaEtal97] A. R. Calderbank, E. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction and orthogonal geometry", *Physical Review Letters* 78 (1997), 405–408.

[Ch84] B. Chazelle, "Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm", *SIAM Journal on Computing* 13 (1984), 488–507.

[Ch91] B. Chazelle, "Triangulating a simple polygon in linear time", *Discrete & Computational Geometry* 6 (1991), 485–524.

[ChPa90] B. Chazelle and L. Palios, "Triangulating a nonconvex polytope", *Discrete & Computational Geometry* 5 (1990), 505–526.

[ChKuRa] D. Cherkashin, A. Kulikov, and A. Raigorodskii, "On the chromatic numbers of small-dimensional Euclidean spaces", *arXiv*:1512.03472.

[ChGoTa95] M. Chrobak, M. T. Goodrich, and R. Tamassia, "On the volume and resolution of 3-dimensional convex graph drawing", in *Electronic Proceedings of the 5th MSI-Stony Brook Workshop on Computational Geometry*, 1995.

[ChGoTa96] M. Chrobak, M. T. Goodrich, and R. Tamassia, "Convex drawings of graphs in two and three dimensions (preliminary version)", *Proceedings of the 12th Annual Symposium on Computational Geometry*, ACM, 1996, 319–328.

[ClEtal90] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl, "Combinatorial complexity bounds for arrangements of curves and spheres", *Discrete & Computational Geometry* 5 (1990), 99–160.

[CoEl03] H. Cohn and N. Elkies, "New upper bounds on sphere packings I", *Annals of Mathematics* 157 (2003), 689–714.

[CoEtal16] H. Cohn, A. Kumar, S. D. Miller, D. Radchenko, and M. Viazovska, "The sphere packing problem in dimension 24", preprint, 2016, arXiv:1603.06518.

[CoZh14] H. Cohn and Y. Zhao, “Sphere packing bounds via spherical codes,” *Duke Mathematics Journal* 163 (2014), 1965–2002.

[CoSl93] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, 1993.

[CsSa93] J. Csima and E. Sawyer, “There exist $6n/13$ ordinary points”, *Discrete & Computational Geometry* 9 (1993), 187–202.

[Cs96] G. Csizmadia, “Furthest neighbors in space”, *Discrete Mathematics* 150 (1996), 81–88.

[Da95] K. Daniels,“Containment algorithms for nonconvex polygons with applications to layout”, Ph.D. thesis, Harvard University, 1995.

[de93] M. de Berg, *Ray Shooting, Depth Orders and Hidden Surface Removal*, Springer-Verlag, 1993.

[DeEd94] T. K. Dey and H. Edelsbrunner, “Counting triangle crossings and halving planes”, *Discrete & Computational Geometry* 12 (1994), 281–289.

[Do94] S. E. Dorward, “A survey of object-space hidden surface removal”, *International Journal of Computational Geometry and Its Applications* 4 (1994), 325–362.

[Ed87] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987. (Monograph on combinatorial and algorithmic aspects of point configurations and hyperplane arrangements.)

[Ed05] H. Edelsbrunner, “Smooth surfaces for multi-scale shape representation”, in *Foundations of Software Technology and Theoretical Computer Science*, P. S. Thiagarajan (ed.), Lecture Notes in Computer Science 1026, Springer, 1996, 391–412.

[EdHa91] H. Edelsbrunner and P. Hajnal, “A lower bound on the number of unit distances between the points of a convex polygon”, *Journal of Combinatorial Theory A* 56 (1991), 312–316.

[EdSk89] H. Edelsbrunner and S. Skiena, “On the number of furthest neighbor pairs in a point set”, *American Mathematical Monthly* 96 (1989), 614–618.

[El67] P. D. T. A. Elliott, “On the number of circles determined by n points”, *Acta Mathematica Academiae Scientiarum Hungaricae* 18 (1967), 181–188.

[Er46] P. Erdős, “On sets of distances of n points”, *American Mathematical Monthly* 53 (1946), 248–250.

[Er60] P. Erdős, “On sets of distances of n points in Euclidean space”, *Magyar Tudományos Akadḿia Közleményei* 5 (1960), 165–169.

[ErFiFü91] P. Erdős, P. Fishburn, and Z. Füredi, “Midpoints of diagonals of convex n-gons”, *SIAM Journal on Discrete Mathematics* 4 (1991), 329–341.

[ErEtal93] P. Erdős, Z. Füredi, J. Pach, and Z. Ruzsa, “The grid revisited”, *Discrete Mathematics* 111 (1993), 189–196.

[ErPa90] P. Erdős and J. Pach, “Variations on the theme of repeated distances”, *Combinatorica* 10 (1990), 261–269.

[ErSz35] P. Erdős and G. Szekeres, “A combinatorial problem in geometry”, *Compositio Mathematica* 2 (1935), 463–470.

[FeKu93] G. Fejes Tóth and W. Kuperberg, “Packing and covering with convex sets”, in *Handbook of Convex Geometry*, P. M. Gruber and J. M. Wills (eds.), North-Holland, 1993, 799–860.

[FeGo17] S. Felsner and J. E. Goodman, *Pseudoline arrangements*, in C. D. Tóth, J. O'Rourke, and J. E. Goodman (eds.), *Handbook of Discrete and Computational Geometry*, 3rd ed., CRC Press, 2017, Chapter 5.

[FeEtal16] C. R. Ferreira, M. V. A. Andrade, S. V. G. Magalhães, and W. R. Franklin, "An efficient external memory algorithm for terrain viewshed computation", *ACM Transactions on Spatial Algorithms and Systems* 2 (2016), article 6.

[Fo93] S. Fortune, "Progress in computational geometry", in *Directions in Geometric Computing*, R. Martin (ed.), Information Geometers Ltd., 1993, 81–128.

[Fo96] S. Fortune, "A beam tracing algorithm for prediction of indoor radio propagation", in *Applied Computational Geometry Towards Geometric Engineering*, M. C. Lin and D. Manocha (eds.), Lecture Notes in Computer Science 1148, Springer, 1996, 37–40.

[FoGaKe93] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press/Wadsworth Publishing Co., 1993.

[FrRö86] P. Frankl and V. Rödl, "All triangles are Ramsey", *Transactions of the American Mathematical Society* 297 (1986), 777–779.

[FrRö90] P. Frankl and V. Rödl, "A partition property of simplices in Euclidean space", *Journal of the American Mathematical Society* 3 (1990), 1–7.

[FrRa94] W. R. Franklin and C. Ray, "Higher isn't necessarily better: visibility algorithms and experiments", in *Advances in GIS Research: 6th International Symposium on Spatial Data Handling*, T. C. Waugh and R. G. Healey (eds.), 1994, 751–770.

[FrEtal94] W. R. Franklin, V. Sivaswami, D. Sun, M. Kankanhalli, and C. Narayanaswami, "Calculating the area of overlaid polygons without constructing the overlay", *Cartography and Geographic Information Systems* 21 (1994), 81–89.

[FrAh84] H. Freeman and J. Ahn, "A system for automatic name placement", *4th Jerusalem Conference on Information Technology (JCIT); Next Decade in Information Technology*, IEEE Computer Society Press, 1984, 134–143.

[Fü90] Z. Füredi, "The maximum number of unit distances in a convex n-gon", *Journal of Combinatorial Theory A* 55 (1990), 316–320.

[Ga77] M. Gardner, "Mathematical recreations", *Scientific American* 236 (Jan. 1977), 110–121.

[GiORSu96] C. Gitlin, J. O'Rourke, and V. Subramanian, "On reconstructing polyhedra from parallel slices", *International Journal of Computational Geometry & Applications* 6 (1996), 103–122.

[GoEtal15] T. L. Gomes, S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and G. C. Pena, "Efficiently computing the drainage network on massive terrains using external memory flooding process", *Geoinformatica* 19 (2015), 671–692.

[GrRoSp90] R. Graham, B. Rothschild, and J. Spencer, *Ramsey Theory*, 2nd ed., Wiley, 1990.

[GrTa13] B. Green and T. Tao, "On sets defining few ordinary lines", *Discrete & Computational Geometry* 50 (2013), 409–468.

[GrWi93] P. M. Gruber and J. M. Wills, eds., *Handbook of Convex Geometry, Vols. A and B*, North Holland, 1993.

[Gr56] B. Grünbaum, "A proof of Vázsonyi's conjecture", *Bulletin of the Research Council of Israel, Section A* 6 (1956), 77–78.

[Gr67] B. Grünbaum, *Convex Polytopes*, Wiley, 1967.

[Gr72] B. Grünbaum, *Arrangements and Spreads*, CBMS Regional Conference Series in Mathematics 10, American Mathematical Society, 1972.

[GrSh90] B. Grünbaum and G. C. Shephard, *Tilings and Patterns*, Freeman, 1990.

[GuEtal95] P. Gupta, R. Janardan, J. Majhi, and T. Woo, "Efficient geometric algorithms for workpiece orientation in 4- and 5-axis NC-machining", *Electronic Proceedings of the 5th MSI-Stony Brook Workshop on Computational Geometry*, 1995.

[GuKa15] L. Guth and N. H. Katz "On the Erdős distinct distances problem in the plane", *Annals of Mathematics* 181 (2015), 155–190.

[Ha94] T. C. Hales, "The status of the Kepler conjecture", *The Mathematical Intelligencer* 16 (1994), 47–58.

[Ha05] T. C. Hales, "A proof of the Kepler conjecture", *Annals of Mathematics* 162 (2005), 1065–1185.

[Ha12] T. C. Hales, *Dense Sphere Packings: A Blueprint for Formal Proofs*, Cambridge University Press, 2012.

[Ha65] S. Hansen, "A generalization of a theorem of Sylvester on lines determined by a finite set", *Mathematica Scandinavica* 16 (1965), 175–180.

[Ha80] S. Hansen, "On configurations in 3-space without elementary planes and on the number of ordinary planes", *Mathematica Scandinavica* 47 (1980), 181–194.

[Ha74] H. Harborth, "Lösung zu problem 664A", *Elemente der Mathematik* 29 (1974), 14–15.

[He23] E. Helly, "Über Mengen konvexer Körper mit gemeinschaftlichen Punkten", *Jahresbericht der Deutschen Mathematiker-Vereinigung* 32 (1923), 175–176.

[Ho96] C. M. Hoffmann, "How solid is solid modeling?" in *Applied Computational Geometry Towards Geometric Engineering*, M. C. Lin and D. Manocha (eds.), Lecture Notes in Computer Science 1148, Springer, 1996, 1–8.

[HoPa34] H. Hopf and E. Pannwitz, "Aufgabe Nr. 167", *Jahresbericht der Deutschen Mathematiker-Vereinigung* 43 (1934), 114.

[IePa95] D. Ierardi and S. Park, "Rigid molecular docking by surface registration at multiple resolutions", *Electronic Proceedings of the 5th MSI-Stony Brook Workshop on Computational Geometry*, 1995.

[Já92] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[Ja87] R. Jamison, "Direction trees", *Discrete & Computational Geometry* 2 (1987), 249–254.

[JeTo95] T. R. Jensen and B. Toft, *Graph Coloring Problems*, Wiley-Interscience, 1995.

[KaLe78] G. A. Kabatiansky and V. I. Levenshtein, "On bounds for packings on a sphere and in space" (in Russian), *Problemy Peredachi Informacii* 14 (1978), 3–25; English translation in *Problems of Information Transmission* 14 (1978), 1–17.

[Ka84] G. Kalai, "Intersection patterns of convex sets", *Israel Journal of Mathematics* 48 (1984), 161–174.

[KaEtal12] H. Kaplan, J. Matoušek, Z. Safernová, and M. Sharir, "Unit distances in three dimensions", *Combinatorics, Probability and Computing* 21 (2012), 597–610.

[KáPaTó98] G. Károlyi, J. Pach, and G. Tóth, “Ramsey-type results for geometric graphs”, *Discrete & Computational Geometry* 20 (1998), 375–388.

[KaWa01] M. Kaufmann and D. Wagner, eds., *Drawing Graphs: Methods and Models*, Lecture Notes in Computer Science 2025, Springer-Verlag, 2001.

[Ki83] D. Kirkpatrick, “Optimal search in planar subdivisions”, *SIAM Journal on Computing* 12 (1983), 28–35.

[Kn92] D. E. Knuth, *Axioms and Hulls*, Lecture Notes in Computer Science 606, Springer-Verlag, 1992.

[Ko93] P. Komjáth, “Set theoretic constructions in Euclidean spaces”, in *New Trends in Discrete and Computational Geometry*, J. Pach (ed.), Springer-Verlag, 1993.

[Kr46] M. A. Krasnoselskiĭ, “Sur un critère pour qu’un domain soit étoilé”, (Russian, with French summary), *Matematicheskiĭ Sbornik, N. S.* 19 (1946), 309–310.

[Ku79] Y. Kupitz, “Extremal Problems in Combinatorial Geometry”, *Aarhus University Lecture Notes Series* 53, Aarhus University, 1979.

[La01] B. Landgraf, “3D graph drawing”, in *Drawing Graphs: Methods and Models*, M. Kaufmann and D. Wagner (eds.), Lecture Notes in Computer Science 2025, Springer-Verlag, 2001, 172–192.

[Le03] F. T. Leighton, *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks*, The MIT Press, 2003.

[LiMi95] Z. Li and V. Milenkovic, “Compaction and separation algorithms for nonconvex polygons and their applications”, *European Journal of Operational Research* 84 (1995), 539–561.

[LiMa96] M. C. Lin and D. Manocha, eds., *Applied Computational Geometry Towards Geometric Engineering*, Lecture Notes in Computer Science 1148, Springer, 1996.

[Lo71] L. Lovász, “On the number of halving lines”, *Annales Universitatis Scientarium Budapest, Eötvös, Sectio Mathematica* 14 (1971), 107–108.

[MaEtal15a] S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and W. Li, “Fast exact parallel map overlay using a two-level uniform grid”, *Proceedings of the 4th International ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, ACM, 2015, 45–54.

[MaEtal15b] S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and W. Li, “Fast path planning under polygonal obstacle constraints”, *4th GIS-focused Algorithm Competition, GISCUP*, 2015, Winner (2nd place).

[MaEtal16] S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and W. Li, “PinMesh – Fast and exact 3D point location queries using a uniform grid”, *Computers & Graphics* 58 (2016), 1–11.

[Ma93] J. Matoušek, “Geometric Range Searching”, Technical Report, *FB Mathematik und Informatik*, Freie Universität Berlin, 1993.

[Mc80] P. McMullen, “Convex bodies which tile space by translation”, *Mathematika* 27 (1980), 113–121.

[MeNä95] K. Mehlhorn and S. Näher, “LEDA: a platform for combinatorial and geometric computing”, *Communications of the ACM* 38 (1995), 96–102.

[Mi96] J. S. B. Mitchell, “On some applications of computational geometry in manufacturing and virtual environments”, in *Applied Computational Geometry Towards*

Geometric Engineering, M. C. Lin and D. Manocha (eds.), Lecture Notes in Computer Science 1148, Springer, 1996, 37–40.

[MiVa92] S. A. Mitchell and S. A. Vavasis, "Quality mesh generation in three dimensions", *Proceedings of the 8th Annual Symposium on Computational Geometry*, ACM, 1992, 212–221.

[Mu94] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, 1994.

[Mu08] O. R. Musin, "The kissing number in four dimensions", *Annals of Mathematics* 168 (2008), 1–32.

[OdSl79] A. M. Odlyzko and N. J. A. Sloane, "New bounds on the number of unit spheres that can touch a unit sphere in n-dimensions", *Journal of Combinatorial Theory A* 26 (1979), 210–214.

[OR87] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[OR93] J. O'Rourke, "Computational Geometry Column 18", *International Journal of Computational Geometry and Its Applications* 3 (1993), 107–113.

[OR94] J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1994.

[Pa93] J. Pach, ed., *New Trends in Discrete and Computational Geometry*, Springer-Verlag, 1993.

[PaAg95] J. Pach and P. K. Agarwal, *Combinatorial Geometry*, Wiley, 1995.

[PaShSz94] J. Pach, F. Shahrokhi, and M. Szegedy, "Applications of crossing numbers", *10th ACM Symposium on Computational Geometry*, 1994, 198–202.

[PaSh90] J. Pach and M. Sharir, "Repeated angles in the plane and related problems", *Journal of Combinatorial Theory A* 59 (1990), 12–22.

[PaSh98] J. Pach and M. Sharir, "On the number of incidences between points and curves", *Combinatorics, Probability and Computing* 7 (1998), 121–127.

[PaStSz92] J. Pach, W. Steiger, and M. Szemerédi, "An upper bound on the number of planar k-sets", *Discrete & Computational Geometry* 7 (1992), 109–123.

[PaTö94] J. Pach and J. Törőcsik, "Some geometric applications of Dilworth's theorem", *Discrete & Computational Geometry* 12 (1994), 1–7.

[PoWe90] R. Pollack and R. Wenger, "Necessary and sufficient conditions for hyperplane transversals", *Combinatorica* 10 (1990), 307–311.

[PrSh85] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[Ra95a] C. Radin, "Aperiodic tilings in higher dimensions", *Proceedings of the American Mathematical Society* 123 (1995), 3543–3548.

[Ra21] J. Radon, "Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten", *Mathematische Annalen* 83 (1921), 113–115.

[Ra95b] V. T. Rajan, "Computational geometry problems in an integrated circuit design and layout tool", *Electronic Proceedings of the 5th MSI-Stony Brook Workshop on Computational Geometry*, 1995.

[Ri96] J. Richter-Gebert, *Realization Spaces of Polytopes*, Lecture Notes in Mathematics 1643, Springer-Verlag, 1996.

[RiZi97] J. Richter-Gebert and G. M. Ziegler, "Oriented matroids", in *Handbook of Discrete and Computational Geometry*, 3rd ed., C. D. Tóth, J. O'Rourke, and J. E. Goodman (eds.), CRC Press, 2017, Chapter 6.

[RuSe92] J. Ruppert and R. Seidel, "On the difficulty of triangulating three-dimensional nonconvex polyhedra", *Discrete & Computational Geometry* 7 (1992), 227–253.

[Sc91] P. Schorn, "Implementing the XYZ GeoBench: a programming environment for geometric algorithms", in *Computational Geometry – Methods, Algorithms and Applications*, Lecture Notes in Computer Science 553, Springer-Verlag, 1991, 187–202.

[ScVa53] K. Schütte and B. L. Van Der Waerden, "Das Problem der dreizehn Kugeln", *Mathematische Annalen* 125 (1953), 325–334.

[ShAg95] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, 1995.

[ShShSo16] M. Sharir, A. Sheffer, and N. Solomon, "Incidences with curves in $\mathcal{R}^d$", *Electronic Journal of Combinatorics* 23 (2016), #P4.16.

[Sh92] T. C. Shermer, "Recent results in art galleries", *Proceedings of the IEEE* 80 (1992), 1384–1399.

[Sk96] V. Skowronski, "Synthesizing tolerances for optimal design using the Taguchi quality loss function", Ph.D. thesis, Rensselaer Polytechnic Institute, 1996.

[So94] A. Soifer, "Six-realizable set x_6", *Geombinatorics* 3 (1994), 140–145.

[So09] A. Soifer, *The Mathematical Coloring Book*, Springer, 2009.

[SpSzTr84] J. Spencer, E. Szemerédi, and W. T. Trotter, "Unit distances in the Euclidean plane", in *Graph Theory and Combinatorics*, B. Bollobás (ed.), Academic Press, 1984, 293–303.

[SrVo93] V. Srinivasan and H. B. Voelcker, eds., *Proceedings of the 1993 International Forum on Dimensional Tolerancing and Metrology*, American Society of Mechanical Engineers, Center for Research and Technology Development, and Council on Codes and Standards, 1993.

[StBrEa95] P. Stucki, J. Bresenham, and R. Earnshaw, eds., "Rapid prototyping technology", *IEEE Computer Graphics and Applications* 15 (1995), 17–55.

[Su17] A. Suk, "On the Erdős-Szekeres convex polygon problem", *Journal of the American Mathematical Society*, 2017.

[Sz97] L. A. Székely, "Crossing numbers and hard Erdős problems in discrete geometry", *Combinatorics, Probability and Computing* 7 (1997), 353–358.

[SzTr83] E. Szemerédi and W. T. Trotter, "Extremal problems in discrete geometry", *Combinatorica* 3 (1983), 381–392.

[Ta14] R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, CRC Press, 2014.

[TaTo95] R. Tamassia and I. G. Tollis, eds., *Graph Drawing*, Lecture Notes in Computer Science 894, Springer-Verlag, 1995.

[TóORGo17] C. D. Tóth, J. O'Rourke, and J. E. Goodman, eds., *Handbook of Discrete and Computational Geometry*, 3rd ed., CRC Press, 2017. (An extensive, comprehensive reference source in discrete and computational geometry.)

[TóVa98] G. Tóth and P. Valtr, "Note on the Erdős-Szekeres theorem", *Discrete & Computational Geometry* 19 (1998), 457–459.

[Tv66] H. Tverberg, "A generalization of Radon's theorem", *Journal of the London Mathematical Society* 41 (1966), 123–128.

[Un82] P. Ungar, "$2N$ noncollinear points determine at least $2N$ directions", *Journal of Combinatorial Theory A* 33 (1982), 343–347.

[Va11] S. Vance, "Improved sphere packing lower bounds from Hurwitz lattices", *Advances in Mathematics* 227 (2011), 2144–2156.

[Ve13] A. Venkatesh, "A note on sphere packings in high dimension," *International Mathematics Research Notices* 7 (2013), 1628–1642.

[Vi16] M. S. Viazovska, "The sphere packing problem in dimension 8," preprint, 2016, arXiv:1603.04246.

[We91] Emo Weltz, "Smallest enclosing disks (balls and ellipsoids)", in *New Results and New Trends in Computer Science*, Lecture Notes in Computer Science 555, Springer-Verlag, 1991, 359–370.

[WeHa11] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed., Pearson, 2011.

[Ya90] F. F. Yao, "Computational geometry", in *Handbook of Theoretical Computer Science, Vol. A*, J. van Leeuwen (ed.), The MIT Press, Elsevier, 1990, Chapter 7.

[Za13] J. Zahl, "An improved bound on the number of point-surface incidences in three dimensions", *Contributions to Discrete Mathematics* 8 (2013), 100–121.

[Zi94] G. M. Ziegler, *Lectures on Polytopes*, Springer-Verlag, 1994.

[Zi98] G. M. Ziegler, "Oriented matroids today", *Electronic Journal of Combinatorics*, Dynamic Survey DS#4, 1998.

[ŽiVr92] R. Živaljević and S. Vrećica, "The colored Tverberg's problem and complexes of injective functions", *Journal of Combinatorial Theory A* 61 (1992), 309–318.

Web Resources:

`http://cs.brown.edu/cgc/` (Center for Geometric Computing, Brown University.)

`http://cs.brown.edu/stc/` (NSF Graphics and Visualization Center.)

`http://dimacs.rutgers.edu/TechnicalReports/TechReports/1994/94-31.ps` (*On the Complexity of Some Basic Problems in Computational Convexity: II. Volume and Mixed Volumes*, DIMACS Technical Report 94-31.)

`http://graphdrawing.org/index.html` (Graph drawing resources.)

`http://jeffe.cs.illinois.edu/compgeom/compgeom.html` (Jeff Erickson's Computational Geometry pages.)

`http://www.algorithmic-solutions.com/leda/index.htm` (LEDA C++ algorithms for geometric computations.)

`http://www-cgrl.cs.mcgill.ca/%7Egodfried/teaching/cg-web.html` (Computational geometry resources.)

`http://www.cs.cmu.edu/~quake/triangle.html` (Triangle: A Two-dimensional Quality Mesh Generator and Delaunay Triangulator.)

`http://www.cs.mcgill.ca/~fukuda/soft/polyfaq/polyfaq.html` (FAQs in polyhedral computation.)

http://www3.cs.stonybrook.edu/~algorith/major_section/1.6.shtml (The Stony Brook Algorithm Repository, which includes Section 1.6 on Computational Geometry.)

http://www.geom.uiuc.edu/apps/quasitiler/about.html (QuasiTiler 3.0.)

http://www.ics.uci.edu/~eppstein/geom.html (David Eppstein's Geometry in Action pages, a collection of many applications of discrete and computational geometry.)

https://polymake.org/doku.php (Polymake, open-source software for research in polyhedral geometry.)

https://www.math.uci.edu/research/mathematical-visualization (Mathematical visualization site.)